Knowledge Management Process Methodology: An Overview

Joseph M. Firestone, Ph.D.

Introduction

The natural evolution of any new area of consulting includes the appearance of methodologies for problem-solving. If Knowledge Management (KM) is the answer to questions about how to capture tacit knowledge, or accelerate innovation, or share knowledge, or resolve cultural incompatibilities in business processes, then this answer is not really operative until we can apply KM knowledge in such a way as to provide practical, concrete solutions for specific business problems. This need naturally gives rise to the development of recipes, guidelines, and procedures for problem solving, called KM process methodologies, and even to particularly comprehensive methodologies called KM "Life Cycle Methodologies." (LCMs)

Life Cycle methodologies are those that specify a linear sequential stepwise process composed of phases that are themselves composed of tasks and activities employing methods. Methods are procedures that detail the "how to's" of arriving at a valued result. A specification of LCM phases might include **Strategic Planning, Analyzing, Designing**, (planning phases) **Implementing, Testing**, (implementing phases) **Monitoring, Evaluating, and Maintaining** (re-implementing the life cycle phases following evaluation).

LCMs may be broader or narrower in scope. The broadest LCMs, in the present context of KM organization-wide programs, are targeted on KM Program implementations. Less broad are project-oriented LCMs, or those targeted on implementing a KM solution for a particular business process. More narrow LCMs focus on developing specific knowledge production processes, for example, benefit estimation for KM initiatives, or estimating ratio scales for prioritizing objects, based on subjective judgements comparing the objects in relation to selected criteria.

This paper addresses the questions of whether in the search for practical KM solutions organizations should employ:

1. LCMs as opposed to iterative/incremental methodologies, and

2. broader scope LCMs as opposed to more modest "mid-range" process methodologies, embedded within a KM methodology architectural/conceptual

framework informed by a foundation of comprehensive knowledge about KM-related theories, metrics, issues, and IT tools.

After answering these questions the paper moves on to propose and describe a type of KM methodology called KM Framework Methodology (KMFM).

What is Knowledge Management Process Methodology?

There are many available definitions of knowledge management [1], but few specifications that bring the definitions a step closer to analysis and measurement. I define KM as human activity that is part of the Knowledge Management Process (KMP) of an agent or collective. This reduces KM to the definition of KMP. And the KMP, in turn, is an ongoing, persistent, purposeful network of interactions among human-based agents through which the participating agents aim at managing (handling, directing, governing, controlling, coordinating, planning, organizing) other agents, components, and activities participating in the basic knowledge processes (knowledge production and knowledge integration), in order to produce a planned, directed, unified whole, producing, maintaining, enhancing, acquiring, and transmitting the enterprise's knowledge base.

In another place [2], I develop the idea that the basic knowledge processes are related to each other in the context of a Knowledge Life Cycle (KLC), that is initiated in response to needs arising from human decision cycles. The above definition is another way of stating the idea that KM is management of the KLC and its outcomes. But the idea of KM still needs further specification.

Let us note first that the KMP is a business process. I break down this process [3] into three high-level task clusters: interpersonal behavior, knowledge processing behavior, and decision-making behavior. Interpersonal behavior may be further categorized into the following task clusters (there are two levels of task clusters in this hierarchy of process components [2, P. 3]):

- Figurehead or ceremonial KM activity (focuses on performing formal KM acts such as signing contracts, attending public functions on behalf of the enterprise's KM process, and representing the KM process to dignitaries visiting the enterprise);
- Leadership (includes hiring, training, motivating, monitoring, and evaluating staff. It also includes persuading non-KM agents within the enterprise of the validity of KM process activities); and
- Building external relationships -- another political activity designed to build status and to cultivate external sources of support for KM.

KM Knowledge processing behavior includes:

- KM knowledge production (different in that it is here that the rules for knowledge production that are used at the level of knowledge processes are specified);
- KM Knowledge Integration (affected by KM knowledge production, and also affects knowledge production activities by stimulating new ones).

Decision making behavior includes:

- Changing knowledge process rules (involves making the decision to change such rules and causing both the new rules and the mandate to use them to be implemented);
- Crisis Handling (e.g., meeting CEO requests for new competitive intelligence in an area of high strategic interest for an enterprise, and directing rapid development of a KM support infrastructure in response to requests from high level executives);
- Allocating Resources (KM support infrastructures, training, professional conferences, salaries for KM staff, funds for new KM programs, etc.);
- Negotiating agreements (with representatives of business processes over levels of effort for KM, the shape of KM programs, the ROI expected of KM activities, etc.).

In brief, the nature of knowledge management is that it is a complex process composed of the above task clusters broken down into task patterns, executed by agents through decision cycles composed of planning, acting, monitoring, and evaluating activities. Further specification of KM, therefore, involves breaking down these task clusters. This has been done elsewhere. [2, Pp. 49-63]

A Knowledge Management Process Methodology is a specification of a normative process directed at achieving a goal within the Knowledge Management domain. The process is composed of activities, tasks, procedures (sequential sets of tasks), tools, method fragments, and methods. When these are combined by the actors implementing them, they are expected to produce the goal of the process. KM Methodology does not have to be structured to support an assumed "full" life cycle. The process need not be linear in character.

Instead, it can be structured as a collection of methods, tools, and procedures for guiding problem-solving with respect to various KM-related issues. It can make provision for feedback loops, concurrent engineering, iterations, and increments. It can make provision for the discretion of actors in application, and for the use of human judgements. It must provide a means for measuring success or failure in attaining the goal(s) of the normative process. The subject matter of the KM domain is specified above, and in the KLC model referred to earlier.

KM Life Cycle Methodology

KM Life Cycle Methodology is a type of KM process methodology that is linear in character. Metaphorically, it follows a "waterfall model," with one task following another, and one phase following another. It does not provide for discretion in its application. It provides no feedback loops, concurrent engineering, iterations, or increments. As with Life Cycle methodologies in general, KM Life Cycle Methodologies include phases such as **Strategic Planning, Analyzing, Designing**, **Implementing, Testing**, **Monitoring, Evaluating, and Maintaining**. But these phases are not executed iteratively, nor deliver KM solutions incrementally. Instead, the implementation model in KM Life Cycle Methodology is "galactic" in nature, attacking every problem as part of the same project, and maximizing the risk in KM implementations.

KM Life Cycle Methodology (KMLCM) Vs, Iterative, Incremental Methodology

Waterfall Life Cycle methodologies have been discredited in various fields. In software development, they have failed to accord with the reality of project work. Such work often does not follow the linear pattern of the waterfall. Iterations occur in the linear life cycle and leave developers hanging. Additionally, it is very difficult for clients to specify all of their requirements at the beginning of a development project. New requirements are always occurring. In iterative development, new requirements can be handled in the next iteration. In the waterfall, a program or project crisis occurs. Further, because of the absence of increments In the waterfall, the KM customer must wait for the completion of a project. Until that time no result is available for use, and there is nothing the customer can evaluate to reinforce commitment to the project.

In discussing software development projects, Alistair Cockburn [4, Pp, 120-121] points out that every project involves a "Validation-V" sequence proceeding from requirements specification and moving through testing. He says that: "*Waterfall* development is when there is only one copy of the V, no matter how long the project is. Incremental development requires having a series of Vs, so that lessons from one V feed the next."

Broader Scope KMLCM vs. Narrower Scope KM Process Methodology

Should we take a broad-scope life cycle approach to KM? We have already answered this question in the negative with respect to the life cycle part of this question. Now we need to consider the question of breadth of scope of KM Process Methodology.

In general, the broader the scope, the more difficult it is to construct a methodology that will provide a useful framework of tasks as opposed to one that will provide a task or activity structure that is devoid of meaningful content. The

reason for this is that KM methodologies having broader scope correspond to multiple step KM and knowledge processes of great complexity. Our knowledge about these processes and how to drive them is highly imperfect. So the steps in methodologies dealing with them cannot be structured and specified in such a way that reliable results of implementing the steps are guaranteed.

Knowing this, authors of broad methodologies specify these steps in a manner that is devoid of content. They order the consumer of the methodology to implement certain steps without actually specifying the procedure for doing it. They leave gaps in the methodology process. The broader the scope of the methodology, the more implementation gaps will exist in it. The consumer buys into the methodology because it is perceived as comprehensive, but the truth is that procedural gaps, caused by gaps in theoretical knowledge, destroy its effectiveness in achieving its goals.

In other words, broad scope KM process methodologies carry with them great risk. And the broader the scope, the greater the risk. The promise of KM process methodology is that it is supposed to guide consumers in implementing practical KM solutions. But broad scope methodology does not deliver on this promise, because its weak foundation in reliable theoretical knowledge delivers risk rather than results at key points in the methodology process.

In contrast, narrow scope methodology focuses on much shorter and less complex value networks. The connection between the methodology's tasks and the intended results is much more direct. The cause and effect interactions postulated by narrow-scope methodologies are much more tightly coupled. As a result, such methodologies are much more likely to deliver results than the much more ambitious broad scope and life cycle methodologies.

KM Framework Methodology (KMFM): Task Patterns, Business Structure Specification, and Iterative/Incremental Development as an Alternative to the KMLCM

If KM life cycle methodologies, and broad scope KM process methodologies, are both either impractical or risky, how should we go about developing solutions to KM problems? What kind of methodology should we pursue? I propose we develop solutions through a type of KM process methodology that specifies task patterns and tasks, knowledge-related business structures, and patterns of iterative/incremental solution development. I call this KM Framework Methodology (KMFM), because it begins with a conceptual framework for developing a KM solution and iterates on that framework until a physical implementation of it is realized.

The application of KMFM, like that of other methodologies, begins with a statement of enterprise goals and objectives and high-level purposes, and

identifies benefit and value gaps between what is and what these goals and objectives envision. It then spells out the problems that need to be solved to close the benefit gaps. It then begins to specify the solution by focusing on the task patterns and tasks, initiated by agents, that the KM system solution must perform to provide an observable result of value to a particular agent.

KMFM is Task Pattern-Driven

KMFM is KM task pattern-driven. A KM task pattern is a set of linked KM activities (defined earlier) initiated by an agent, and performed by the system solution, producing a result of value for a particular agent. KM agents initiate tasks in order to affect KM interaction itself and/or in order to affect the Knowledge Life Cycle or one of its components or outcomes. The result of value, the goal of a task, is an effect of the above sort that the agent expects the system to produce.

Some KM task patterns, such as sequences of interaction with a computer network (often called use cases [5]), are performed on the physical world, and the response of the physical world to the agent, in the form of the computer's response, is determined by cause and effect. But many more KM task patterns, such as hiring new knowledge workers, are social in character, involve working and collaborating with people, and even though their outcomes may often be somewhat predictable, they are not determinate, as are the outcomes of tasks performed on a computer.

Moreover, KM task patterns that involve interaction with people and groups are also unlike use cases in that the agents performing them are not external to the knowledge life cycle as people are external to the information system. Instead, they are part of the organization, the knowledge life cycle, and the KM processes that they act upon,

For these reason it would be inappropriate to refer to KM task patterns, in general, as "use cases," as that term is used in object technology [5] and specifically in the Unified Modeling Language (UML). [6] Most KM task patterns are different from computer use cases, in that they don't have determinate responses, and in that the agents performing them are "part of the process" they are influencing. Action and effect are not determined through a causal chain that we either understand or control.

Nevertheless, KM task patterns in KMFM are also similar to use cases in that they are (a) KM and knowledge worker-centric (focused on the user) and (b) goal-oriented (intended to produce particular valued outcomes). Further, the set of KM task patterns in a methodology is like the set of use cases in a software application, in that the tasks and their expected outcomes describe the functionality that KM-workers *intend* to build into their KM system solution. KMFM

development, including its phases, iterations, and increments, is focused on (and organized around) constructing a solution that will implement such task patterns. It is in this sense that we can say that KMFM is task-driven.

KMFM is Business Structure-Centric

The second major characteristic of KMFM is that it is business structure-centric. "Business structure" here refers to the specification of a model of business objects and relationships incorporating organizational knowledge. The organization behaves through its business structures including: normative business processes, strategic plans, authority structures, information systems, policies and procedures, etc. Organizational knowledge exists within these business structures, in particular configurations found in them.

Sidebar One -- Some Business Structures of the Enterprise

▪ Business Processes
▪ Organizational Culture
▪ Organizational Strategy
▪ Organizational Teams
▪ Formal Or. Sub-divisions
▪ Individuals
▪ Policies
▪ Procedures
	▪ Products
	▪ Services
	▪ Codified Organizational Knowledge
	  ▪ Information Systems
	  ▪ Paper Documents
	  ▪ Images
	  ▪ Art
	  ▪ Other Organizational Cultural Artifacts

Again, business structure models describe the objects and relationships through which task patterns and tasks are performed. Agents initiate task patterns using the existing business structure to interact with other agents, The combination of tasks, structure and responses of other agents is indeterminate in terms of cause and effect. Nevertheless, the solutions produced through KMFM are combinations of tasks that leverage business structures to produce behavior aligned with self-organizing tendencies toward knowledge growth and innovation.

In addition to producing tasks then, KMFM requires development of a business structure model of the objects and relationships that support the execution of task patterns in KM and knowledge processing interactions. The relationship between task pattern and business structure specification is one of continuous alternation. We develop task patterns a little to express desired functionality in the KM solution, and then specify the business structure a little that supports these task patterns. The alternating pattern of iterative/incremental development is the next central characteristic of KMFM.

Iterative/Incremental Development

An iteration is an ordered (though not linear, and not determinate) collection of goal-directed activities. It constitutes a portion of a solutions development process, organized in a work flow that produces a valued outcome. An increment is the valued outcome produced by an iteration. In KMFM, following Rational Software's Unified Software Development Process [7], a project cycle directed at a solution is divided into phases, and the phases into a series of iterations, or mini-projects, each producing an increment. When KMFM is working well (and realistically, no methodology works well all of the time) each additional iteration produces an increment of the solution that is closer to the project goal.

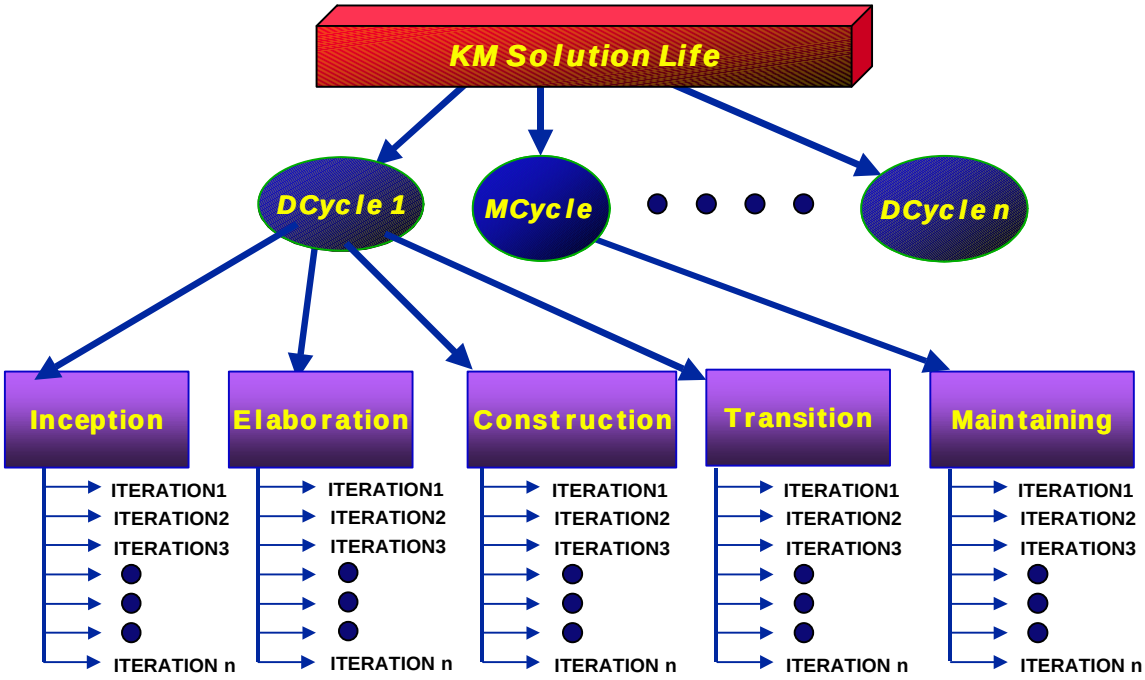

Figure One --The KM Project/Cycle/Phase/Iteration Hierarchy

Iterations are used in KMFM to reduce risk in development. By designing iterations carefully, and by successfully completing them, KM solutions developers can:

- expose risks in KM projects in early iterations, and by successfully completing these iterations, can mitigate and, in some cases, even remove these risks; and
- produce increments that deliver immediate value to agents looking for KM solutions.

Iterations occur in the context of the Life Cycle of a KM solution. The Life Cycle is itself divided into development project cycles and maintenance cycles. Each of the development cycles is itself divided into four phases: **Inception, Elaboration,**

Construction, and Transition. [7, Pp. 11-13 and Part III] Each maintenance cycle has a single Maintenance and Enhancement phase. In turn, each phase may be further divided into iterations producing increments, and each of these iterations may be divided into work flows. The four phases of development cycles are:

- *Inception* is defined as the first phase of a project cycle in which the basic idea of the solutions development project, including the most important task patterns within it, is developed sufficiently to make an initial business case to justify moving into the elaboration phase.

- *Elaboration* is the phase of an iteration in which the architecture of the solution (i.e. the to-be configuration of business structures that is projected by the solution to impact the system so as to produce the valued outcome projected by the solution) is defined and base-lined. In addition, task patterns and tasks are specified in great detail.

- *Construction* is the phase where the solution is developed from architecture to the actual business structure configuration specified by the solution, but construction may also involve creation of new task patterns and further development of architecture.

- *Transition* is the phase where the KM solution is tested, evaluated, and deployed by the agent community.

The phases in a cycle in KMFM may, again following the Rational Unified Process (RUP) [7], be viewed from the viewpoint of the "core work flows" completed in any iteration within a phase. These work flows (similar but not the same as in RUP) are: **Strategic Planning, Analyzing, Designing**, **Implementing, Testing (including Monitoring and Evaluating), and Maintaining.** They are similar to the typical phases in the KMLCM. This suggests that using iterations in KMFM, in contrast to KMLCM, requires de-coupling, and clearly distinguishing the difference between the phase and work flow concepts in methodology.

The phase concept relates to time segments within cycles, and to the presence of milestones at the end of each segment. On the other hand, the work flow concept is concerned with the kind of work that may or may not occur in each iteration within each phase. Since work flow is distinguished from phasing, KMFM allows one to describe the extent to which any phase of a cycle and any iteration within that phase, includes multiple or even all work flows, or even more flexibly, involves all work flows but with differing degrees of emphasis.

Through iterative and incremental development, KMFM provides an architecturally rich value network for solutions development. Iterations produce

increments throughout the phases of a KM solution project cycle. But increments are not produced through linear life cycle development. Instead iterations occur within phases based on time, and work flows are not co-extensive with these phases. Task pattern specification (analysis) can occur in any phase of an iteration, as can structural modeling (designing), strategic planning, or any of the other work flows. Iterations can be implemented in parallel as well, if that is what is called for to meet certain deadlines.

The pattern of iterative development is not driven by phasing in a linear life cycle, but by the logic of implementing selected prioritized task patterns through iterations, in order to meet needs specified in a projected increment. As iterations are implemented, increments gradually add task patterns to those produced by earlier increments. KM solutions are gradually enhanced over time, in an orderly way, informed by prioritization of task patterns introduced in the strategic planning work flows of various iterations.

Work Flows In KMFM

Again, the work flows in iterations are **Strategic Planning, Analyzing, Designing**, **Implementing, Testing (including Monitoring and Evaluating), and Maintaining.** Here are brief descriptions of what is involved in these various work flows.

Strategic Planning (and Requirements Capture)

Step 1: Strategic planning and requirements capture begins with **constructing a model of the goals and objectives of the enterprise and the target business process.** These impact on and are logically prior to its business processes, including knowledge management, in the sense that the processes are value networks whose purpose is to accomplish these goals and objectives. The strategic and tactical goals and objectives may be expressed as an analytic hierarchy [8], a tree structure with strategic goals at the highest level, tactical goals at the next lowest level, and objectives below the tactical goals.

Once an analytic hierarchy has been constructed, a foundation exists for evaluating the "as-is" and "to-be" knowledge and KM processes, and their component task patterns in terms of the "gap" between their results and the goals and objectives in the hierarchy. So development of the analytic hierarchy at the beginning of this work flow is critical in establishing a baseline for evaluating either predicted or actual results of a KM program or initiative.

Step 2: The next step in the KMFM process is **to analyze the "as-is" knowledge and KM processes and their outcomes.** That is, one needs to do a "KM audit." To do this you need to adopt or build a conceptual model of knowledge and KM processes, and of knowledge outcomes (as well as

knowledge management metrics based on the framework) that can frame the audit. You'll find such a framework in some earlier publications of mine [2] [9] [10] and others [11] [12] presenting the KMCI Knowledge Life Cycle (KLC), and Metaprise models [13] [14]. These frameworks define KLC and KM concepts, specify and characterize them in terms of concepts and metrics, and begin to relate these concepts and metrics in a semantic network. Other frameworks may be used as well. The point here is that some framework, either explicit or implicit, must ground the analysis. I think it is better to be explicit about the framework, so that it can serve as the foundation for future performance measurement. And this means that you must either adopt a framework or build one in performing this step.

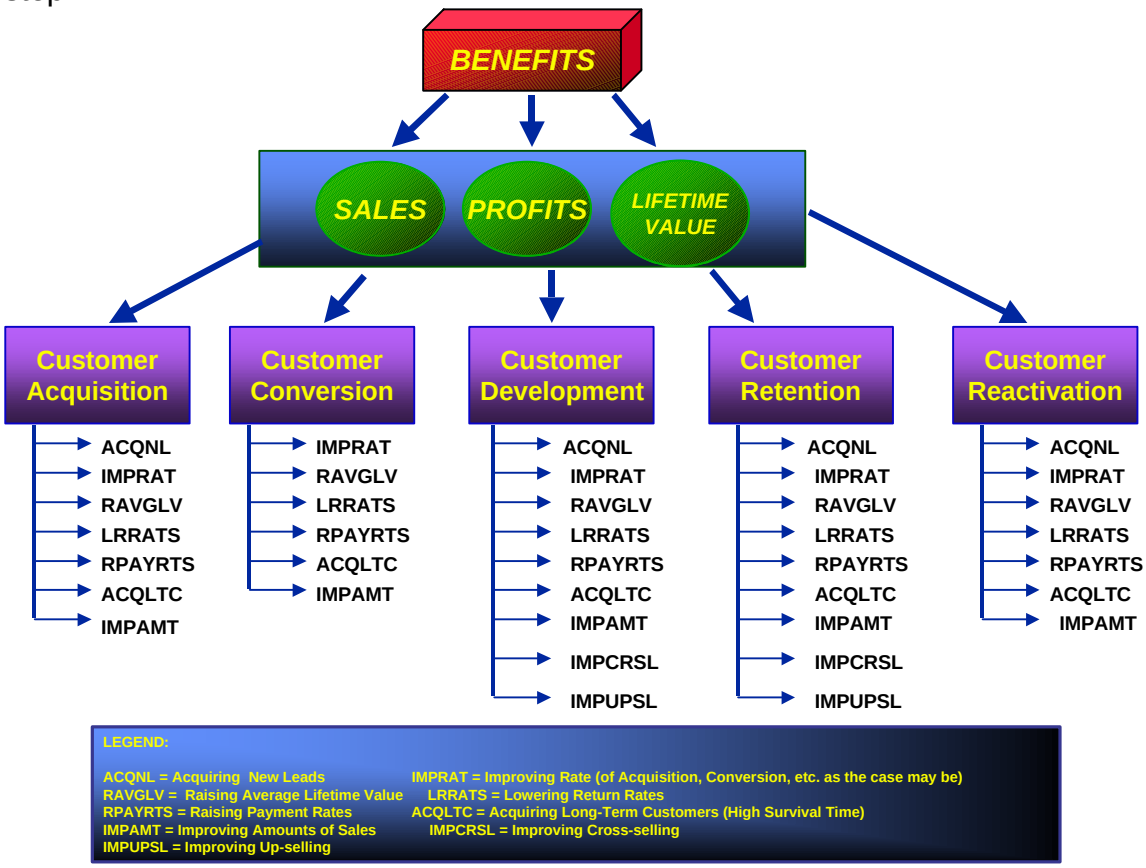

Figure Two -- An Analytic Hierarchy

Step 3: Take the outcome of the "as-is" process and map it onto the hierarchy of goals and objectives. There are two aspects of such a mapping: the reality gap between goals and objectives and the "as-is" model, and the valuational or benefit gap between the two. [15] It is the benefit gap that is more important, since it takes into account the value we place on the actual gap found using empirical measures. I have outlined a method for mapping an "as-is" condition onto an analytic hierarchy in [15].

Step 4: Analyze the problems that need to be solved to close the benefit gap and select the first iteration of the KM solution. "Problems" here refers to knowing what to do to change knowledge and KM processing to create outcomes which close the benefit gap. So "analyzing the problems" means developing models that explain how the changes we are looking for may be induced. McElroy [16] provides an example that illustrates at least part of what I have in mind in his analysis of what it takes to accelerate innovation in the enterprise. But his analysis is a conceptual model addressed toward an abstract goal, while I have in mind mathematical or computer modeling addressed to the range of goals and objectives defining the benefit gap. McElroy [17] developed a prototype system dynamics simulation for the MacroInnovation Associates web site, but this is a suggestive illustration, rather than the full-blown model needed to analyze the impact of KM solutions on innovation.

How rigorous and wide ranging the analysis of problems is at this stage will depend on what an organization is willing to fund. The more rigorous the analysis, the better the position one is in to define an iteration of the solution that will be practical to implement in a short time following the end of the Strategic Planning work flow. But the trade-off is that getting the knowledge that will enable a good decision takes more work and expense during this work flow.

Step 5: Specify the initial solution by focusing on the task patterns and tasks, initiated by agents, that the iteration of the KM system solution must perform to provide an observable result of value to a particular agent. The tasks and task patterns needed to arrive at a solution are the products of this step in the work flow. A linked sequence of activities performed by one or more agents sharing at least one objective is a ***task***. A linked, but not necessarily sequential set of tasks governed by rule sets, producing results of measurable value to the agent or agents performing the tasks, is a ***task pattern***. The tasks and task patterns provide the functional requirements for the KM system.

Both tasks and task patterns are specified from *the viewpoint of the agent* and the value the agent expects to get out of initiating and contributing to tasks and task patterns. The solution is structured in this way to maintain accountability to the enterprise knowledge workers who have the responsibility for implementing KM solutions. The task patterns drive the KM solution in the sense that they are what the agents think needs to be done to solve problems and close the benefit gap.

Because tasks and task patterns are tied to agents, the agents must be identified and described before the tasks and task patterns are specified. Task patterns and tasks are then specified through careful description of the course of tasks and task patterns, as agents and objects in the system interact. Description of these courses will typically involve alternative scenarios and may also specify

emergent Complex Adaptive Systems (*cas*) [18] phenomena that are anticipated but not strictly predictable.

Once the individual tasks and task patterns are specified, a model of the task pattern configuration viewed as a whole must be developed. This model focuses on relationships between and among task patterns and agents.

This description of how tasks and task patterns are specified looks similar to use cases [5] in object-oriented software development. But tasks and task patterns are not the same as low-level and high-level use cases, because they don't specify a determinate course of events that is sure to yield a specified result. In KM systems, there is always an indeterminate element in tasks, provided that they involve sequences requiring human and organizational responses, as opposed to information system responses. This indeterminacy makes KM system building much more challenging than software development.

Step 6: Prioritize Task Patterns. Not all task patterns are of equal importance to the iteration. Indeed, the iteration defined in step 4 should be redefined again here in light of how the task patterns have been specified. This is an important risk-reducing step for the project. The task patterns that define the iteration should promise to solve significant problems, but should also be manageable in the time frame available for the iteration.

Step 7: Detail the Task Patterns. Here the courses of the tasks and task patterns are structured and described in much greater detail. The task and task pattern models are visualized and developed graphically and in detailed textual form. Where task patterns are information system tasks, that is, actual use cases, these may be prototyped at this stage to develop a still more specific view of them and to illustrate aspects of the system for users.

Step 8: Structure the Task Patterns and extend them. In this step tasks and task patterns are analyzed to see if they can be structured into higher level patterns of shared functionality. The analysis may also suggest new task patterns that extend system functionality.

Step 9: Specify Supplementary Requirements not embodied in a specific Task or Task Pattern.

Outcomes:

- **Goals and Objectives Model targeted on Knowledge and KM Processing and Outcomes,**
- **An "as-is" model of Knowledge and KM Processes and Outcomes,**
- **Reality and Benefit Gap Models,**
- **An initial Problem Specification and Solutions Model,**

- **A Choice of Iteration,**
- **A Task Pattern Model specifying Functional Requirements and Prioritization of Task Patterns,**
- **A Revision of the Iteration, and**
- **A list of Supplementary Requirements.**

Analyzing

This work flow analyzes the classes of objects that perform task patterns (the analysis classes) and the requirements of the "to-be" system as specified in task pattern realizations. As in UML-based methodologies, there are three types of analysis classes in KMFM: boundary, entity, and control classes. [19] Boundary classes are those used to model objects that manage inputs to, and outputs from, an organization. Control classes are used to model objects that coordinate object interactions in one or more task patterns or tasks. Entity classes model objects that manage data, information, and knowledge resources, and access to them within the organization.

Step 1: Specifying The Boundary, Control, and Entity Classes Needed To Express Task Pattern and Task Pattern Realizations. Examples of boundary classes include human agents themselves, whose roles involve communication with customers, regulators, and other agents external to the enterprise, electronic interfaces to external organizations, even doors and windows. Examples of control classes also include human agents. In this case they are human agents who control transactions and interactions within the enterprise, including knowledge managers, data miners, financial analysts, but also any human agents performing tasks. Information systems are other examples of control objects as they control part of the processing of data, information, and knowledge. Examples of entity classes are employees, databases and their component objects, text bases, Invoices, purchase orders, and image stores.

The three types of analysis classes are specified by starting with the requirements captured in the Strategic Planning and Requirements Capture work flow, and particularly with the task patterns and problem solution descriptions. Working from these, the relevant boundary, control, and entity classes, their behavior, their methods, and their attributes are specified.

This specification requires more detailed concept formation and measurement modeling. Concept formation is the process of developing relationships among concepts (constructing a semantic network, concept map, or cognitive map). Measurement modeling is a specific concept formation activity that relates abstract concepts to experiential concepts. [20] [21] [22] [23] The methods in boundary, control, and especially entity classes will need to include an initial specification of measurement models, along with other methods. When concept

formation is restricted to a single boundary, control, or entity class, I call it intra-class concept formation.

Step 2: Analyzing Requirements in Task Pattern Realizations. Analysis also refers to the further refinement of previously captured requirements, by expressing them in object models rather than in task patterns. Each object model "realizes" [5] a task pattern. In these realizations we specify the classes, and the structure of relationships among them, that support the task patterns and produce outcomes initiated by them. This pattern is a structural model of the objects performing task patterns, and is another instance of concept or semantic network formation -- this time interclass concept formation. We also specify how interacting objects collaborate to produce behavior. We perform these specifications with class diagrams, interaction diagrams, and flow of events textual analyses. We also specify special requirements through textual descriptions. Collaboration analysis begins causal modeling in the analysis work flow.

Both structural modeling and causal modeling have long histories in the social sciences and in organizational analysis. [24] [25] [26] [27] These activities are important in KMFM. But they are cast within the over-all framework of systems thinking [28] and object modeling [29] guiding the methodology. So both structural modeling (including measurement modeling), and causal modeling, occur here in the context of object modeling of task pattern realizations.

There are four sources of object models produced by analysis in previous work: (1) the goals and objectives specified earlier, (2) the initial problem-solution model, (3) the task patterns model, and (4) the boundary, entity, and control classes found in step 1. All four of these help to identify the objects, their attributes, their methods, and their resulting behavior. In addition to the structural relationships emphasized in object models of software applications, models of KM and other human system solutions must include possible measurement and cause-effect relations between pairs of objects. Once again, these are among the most important relations in the object model.

Step 3: Integrating Task Pattern Realizations In a Single Object Model

Entity, control, and boundary objects frequently support more than one task pattern. So, object models overlap. This overlap creates the opportunity to integrate these models (task pattern realizations) into a global object model supporting all task patterns constituting the KM solution.

Step 4: Using the Analysis Model, Refine the Initial Specification of the KM Solution and Develop Alternative Object Models and KM Solutions. The initial KM solution specified in the Strategic Planning work flow, is refined using the knowledge gained in the analysis work flow. In addition, alternative object

models and KM solutions are formulated for testing the primary KM solution later. This requires developing alternative analysis classes, structural object models, and object interaction models, and well as configurations of KM task patterns. A much more concrete view of the developing KM solution is thereby gained.

Outcomes:

- **A Set of Boundary, Control, and Entity Classes to be used in Modeling the Solution,**
- **A Task Pattern Realization Object Model specifying Structure and Dynamics of the Solution including measurement and causal models,**
- **A Refined Problem Specification and Solutions Model, and**
- **Alternative Object Models and KM Solutions**

Designing

In the design work flow, the classes, objects, related task pattern realizations and object models are refined beyond the relatively abstract result of the analysis work flow to create corresponding design classes, task pattern realizations and refined object models, both primary and alternative. The refinements in design require specificity sufficient to guide implementation. That specificity requires modeling not only conceptual relations and interactions among objects, but also the physical component of the objects' relations and interactions.

Step 1: Specify design classes. The analysis work flow focuses on developing a clear specification of attributes, operations, and methods of classes. The design work flow takes these analysis classes expressed in language and diagrams and specifies them in the form of mathematical and computer models expressed in physical code and simulations. The design work flow also specifies classes expressing physical details about the enterprise, and classes expressing measurement models [2] used to compute metrics. Since the analysis classes include the alternative classes specified in analysis, the design classes will include design versions of these alternative classes characterized by alternative measurement models and/or key concepts.

Step 2: Specify design object model. Analysis classes have structural relations including associations, generalizations, and dependencies. In the design work flow, structural relations between design classes are specified by giving them a physical implementation in mathematical models and code. It is here that relationships between key conceptual nodes in the business domain are made precise and are programmed. Alternative structural models are also formulated in this step.

Step 3: Specify design object interactions. Here too, what is expressed in analysis through words and diagrams is given a physical implementation in

mathematical models and code. This is the step in which cause-and-effect relations and system dynamics are converted from diagrams and visual models to program code. Alternative cause-and-effect relations and system dynamics are also formulated in this step.

Step 4: Specify implementation requirements. In this step all requirements for implementing the KM solution, both conceptual and physical are specified and are factored into the object model.

Step 5: Test the Design Model. Here testing of the design model occurs. First, the model is checked for apparent logical flaws by the project team. Second, after correcting any errors uncovered, the design model is unit tested, integration tested and finally sensitivity and stress tested through computer simulations. Third, program code is revised as necessary depending on the results of testing. Fourth, alternative design object models are tested in the same manner as the primary object model. And fifth, all design object models are tested against one another for plausibility.

Step 6: Use the Tested Design Models to Develop and Test Refined KM Solutions. That is, take the primary design model and the alternatives, and formulate strategy and tactics for achieving desired goals and objectives. Test the alternative KM solutions through simulations.

Outcomes:

- **A Set of Physical Boundary, Control, and Entity Classes to be used in Modeling the Solution,**
- **A set of Physical Implementation Requirements,**
- **Tested Alternative Task Pattern Realization Object Models specifying Structure and Dynamics of the Models and expressed in physical code, and**
- **Refined and Tested Alternative Problem Specifications and Solution Models.**

Implementing

Implementing is acting in accordance with the expectations set by a design model of task pattern realizations. Implementing may involve a pilot implementation of a specified solution, a partial implementation, or a more comprehensive implementation of a solution being iterated. The nature of the implementation -- its breadth and scope -- will depend on how the iteration is defined in the inception phase of the project.

Implementing a KM solution involves taking action to execute policies expressed in a design model. Earlier I presented a categorization of activities in the KMP.

Implementing is executing some combination of figurehead, building external relations, leadership, KM knowledge production, KM knowledge integration, changing knowledge processing rules, crisis handling, resource allocation, and negotiating resources activities corresponding to strategies and tactics composed of combinations of the task patterns specified in a design model. Implementing will also involve responding to reactions of other agents in and external to the organization, using the task patterns specified in the model.

Among these actions will be implementing *information technology* applications necessary to support the KM solution called for in the model. *Implementing IT applications involves implementing software development methodology*. KMFM prescribes some form of O-O-based methodology such as the Rational Unified Process [7], Extreme Programming Methodology [30], or Cockburn's [32]. All three are consistent with KMFM in that they are iterative and incremental.

Step 1: Select the Best Validated Design Model and Associated KM Solution Resulting from the Design Work Flow

Step 2: Implement IT applications required for the KM Design Solution

Step 3: Implement the Policies Specified in the KM Design Solution

Outcomes:

- **Implemented KM IT Applications**
- **An Implemented Solution**

Testing

In the testing work flow, the effects of Implementing the solution are monitored and evaluated.

Monitoring

In monitoring, the operation and impact of the KM solution is observed and measured. Monitoring is done by comparing object model predictions with events as they play out. The objectives of the comparisons performed in this work flow are description and analysis of occurrences and measurement of the impact of the KM solution, but not assessment of what has happened as a result of implementing it. Monitoring proceeds by using all models, both primary and alternative, to analyze the results of an implementation. It tests both the primary model and its alternatives against events.

Step 1: Compare the Actual Task Patterns and Task Behavior With Task Patterns Prescribed by the KM Solution and Pre-existing Task Patterns before Implementation of the KM Solution.

Step 2: Compare the Changes Associated with Implementing the Task Patterns with the Changes Predicted by the alternative KM Solution Models in the Knowledge Management Process, Its Sub-processes, and Its KM Level Outcomes.

Step 3: Compare the Changes Associated with Implementing the Task Patterns with the Changes Predicted by the alternative KM Solution Models in the Knowledge Life Cycle, Its Sub-processes, and KLC Outcomes.

Step 4: Compare the Changes Associated with Implementing the Task Patterns with the Changes Predicted by the alternative KM Solution Models in other Enterprise Processes, Sub-processes, and Outcomes.

Evaluating [See 15]

Step 1: Evaluate the benefit gap between the Actual Task Patterns and Task Behavior, and Task Patterns Prescribed by the KM Solution, and also the benefit gaps between each of these and the Pre-existing Task Patterns before the KM Solution was Implemented

Step 2: Evaluate the benefit gap between the Changes Associated with Implementing the Task Patterns and the Changes Predicted by the alternative KM Solution Models in the Knowledge Management Process, Its Sub-processes, and Its KM Level Outcomes.

Step 3: Evaluate the benefit gap between the Changes Associated with Implementing the Task Patterns and the Changes Predicted by the alternative KM Solution Models in the Knowledge Life Cycle, Its Sub-processes, and KLC Outcomes.

Step 4: Evaluate the benefit gap between the Changes Associated with Implementing the Task Patterns and the Changes Predicted by the alternative KM Solution Models in other Enterprise Processes, Sub-processes, and Outcomes.

Outcomes:

- **Validated Alternative Object Models and KM Solutions**

Maintaining or Enhancing the KM Solution

The monitoring and evaluating work flows are continuous tests of the validity of the KM solution. The experience resulting from these work flows is applied in reformulating the Task Pattern realizations representing the KM solution.

Step 1: Use the results of Testing (Monitoring and Evaluating) to redesign the KM Solution by Iterating over the design work flow.

Step 2: Change the KM Solution to Correspond to the Redesigned Model and Solution.

Step 3: Implement the Redesigned Solution.

Step 4: Test the Redesigned Solution

Step 5: Go Back to Step 1:

Outcome:

- **Continuous Improvement in KM Solutions**

Phases

The following description of phases relies heavily on the account given by Jacobson, Booch and Rumbaugh, for the Rational Unified Process (RUP), supplemented with a maintenance phase, not included in RUP. [7, Pp. 315-407] The description is summarized and also modified where that is relevant, because KMFM is not a software development methodology. Even though it is not, the RUP account of phases is useful as a basis for discussion. As we shall see it is easy to use as an outline to integrate the necessary work flows and activities for KM solutions development. Another advantage, is that construction or integration of a software application is frequently necessary in constructing a KM solution. In that case the RUP is directly applicable, so that the methodology gap between KM solution and KM software implementation is minimized.

KM solutions evolve in a number of development and maintenance cycles. Each development cycle is divided into four phases: inception, elaboration, construction, and transition. Maintenance cycles have only a single phase. The development cycle phases have already been defined. In this section they will be described in greater detail and the relationships among phases, iterations and work flows will be explained.

Phases don't overlap. They are executed by using at least one iteration, which begins with a planning work flow. They end with milestones completed, and with assessment of the final iteration in the phase, and the phase itself. In between planning and assessment, each of the work flows that constitute each iteration or

"mini-project," are performed. I have described the work flows that can be used in each iteration above. But it is critical to note that each iteration is neither associated entirely with any one work flow, nor need each iteration include all of the work flows, or include all of the work flows to the same degree.

Instead, we vary the selection of work flows according to the objectives of iterations within each phase and the overall objectives of that phase. I will describe each of the phases, the nature of the iterations used to complete them, and the representation of work flows in these iterations and phases.

Inception

Again, inception is defined as the first phase of a project cycle in which the basic idea of the KM solutions development project, including the most important task patterns within it, is developed sufficiently to make an initial business case to justify moving into the elaboration phase. If the initial business case is not made the project is reconsidered and should be abandoned if it can't be reformulated in terms that produce a satisfactory business case.

To make the business case we need to find and specify those task patterns that are critical to the business case. Typically these are only some, and perhaps only a small percentage of the task patterns that will eventually be part of the KM solution. Other things we need to do to make the initial business case are:

- Determine the scope of the system the task patterns will have an impact on. Specifically, model the system boundary and the receptors and effectors it uses to interact with other systems and the environment generally.

- Model the "to-be" system including the KM solution we expect to implement. The model should not be a detailed construct for this phase. It is only a first approximation of the eventual modeling of the system. But it must present enough of a vision of the system so that critics can believe that the solution can be implemented and that it has a good chance of producing the intended effects

- Determine the critical risks that affect feasibility of the system and analyze whether they can be mitigated either in the inception phase or later in the cycle.

- Depending on the nature of the KM solution, it may be necessary to prototype the solution at this stage to demonstrate the feasibility of some critical aspect. This is more likely to be the case if the solution requires an IT application. If it does not it still may be necessary or desirable to build a basic simulation model that demonstrates some of the critical dynamics of the proposed solution.

To accomplish the above we need to perform the following activities.

(1) Consider previous work done in the organization that has produced useful information for your new project. The amount and relevance of such work will vary widely, but other KM systems that have been developed, structured data resources, work on communities of practice, various IT applications, all may be relevant to how much effort is necessary in various phases of the project.

(2) Plan the inception phase. Often this is difficult because at inception there may be little available information structured in a way that is relevant to the project. You need to gather all previous information that may be relevant, structure it so you can use it, locate people who can work with this information, and find out what information is missing that is needed to determine system scope, model the system, analyze risk, or do a prototype.

(3) Develop the vision of the KM solution. Any project begins with a vague idea of what the problem is and what its solution should accomplish. This vague idea needs to be developed into a vision of the solution that is specific enough to guide the inception phase. Development of this vision requires that stakeholders in the solution come together to exchange views and to synthesize the best vision they can arrive at. The vision must be that of an integrated solution, not one that simply reflects an aggregation of what stakeholders want. What stakeholders want is important, but the vision must also incorporate ideas about what kind of solution will work.

(4) Specify the evaluation criteria for the inception phase. This task results in different criteria depending on the project. But following are some general criteria that will apply to all projects. First, the boundary of the system that is the target for the KM solution must be clearly specified. In particular, it must be possible to clearly distinguish who is in the system, who is outside of it, and whether the system as characterized is likely to be able to function. Second, the task pattern requirements and supplementary requirements needed to make a persuasive business case for the project must be specified. Third, a candidate KM solution along with its architecture must be specified in the inception phase. Then this solution and architecture must be evaluated for plausibility and for the value it delivers to agents. Fourth, one must evaluate whether all the critical risks in the project have been identified, and whether there is a plan for mitigating them. And finally, one must evaluate whether the business case is strong enough to go ahead with the KM solutions project.

(5) Execute the core workflows of the iteration. The inception phase iteration(s) will focus heavily on strategic planning and requirements capture, and less so but still substantially on analysis and design. Little attention will be placed on

other work flow activities such as implementation and testing in the inception phase, unless the solution or some aspect of it must be prototyped to make the business case. In requirements capture activities include: listing requirements for the feature list of the KM solution, gaining an understanding of the enterprise context of the solution, capturing the functional requirements necessary to make the business case as task patterns (find agents and task patterns, prioritize task patterns, detailing some fraction of the task patterns to get a handle on the architecture of the solution), and capturing the non-functional requirements necessary for the business case.

In analysis, in the inception phase, an analysis model is constructed sufficient to support the related tasks of specifying task patterns and the solutions architecture. This involves architectural analysis and detailed analysis of some of the critical task patterns identified in requirements capture. The analysis marks the beginning of an attempt to produce a structural model of the system responding to the task patterns. One objective in the inception phase is to reveal those structures in the enterprise that are shared by the task patterns. Potential conflicts occur here in cases where the capacity of business structures may be strained by their need to process too many, or overly complex task patterns of the KM solution. In inception phase analysis one tries to identify these conflicts and anticipate their solution.

In the design work flow, an initial design model is constructed to tie down the physical architecture of the KM solution in certain critical areas touched on by the critical task patterns analyzed in the inception phase. Little design work is done in these functional areas or in the areas of non-functional requirements. Only enough is done to mitigate risks and support the business case. Finally, little implementing and testing is done unless a demonstration prototype must be constructed. Even in that case only enough is done to demonstrate a critical task pattern.

(6) Construct the business case. At this point enough facts have been gathered and analysis completed to allow one to construct the business case for the KM solution. The inception phase is too short and lacking in detail to develop a precise business case. Nevertheless, one can develop estimates of the resources required by the project, its anticipated benefits and costs (both monetary and non-monetary and its anticipated benefits. These should be organized in a business plan for the project that lays our revenue projections, costs, and a marketing plan for the KM solution.

(7) Assess the inception phase iteration. The assessment considers whether the task patterns defined lay out enough of the solution to lay out the business case. It also considers whether all of the critical risks have been found and adequately mitigated. If the assessment indicates that the objectives of the inception phase have not been achieved, a new iteration is planned to

complete the inception phase. A second iteration is needed only infrequently to complete the inception phase. A third iteration is almost never necessary before a decision to undertake or shelve the project can be made.

(8) Plan the elaboration phase. Plan the activities, costs, and schedule of the elaboration phase. The plan is constructed to provide for identification of most of the task patterns, description of most of them and analysis of at least half of them.

Elaboration

Again*, elaboration* is the phase of a project cycle in which the architecture of the solution (i.e. the to-be configuration of business structures that is projected by the solution to impact the system so as to produce the valued outcome projected by the solution) is defined and base-lined. In addition, task patterns and tasks are specified in great detail. Objectives of the elaboration phase follow. [7, 359-360]

- To finish identifying most of the task patterns.
- To establish an architectural baseline of necessary business structures To guide construction and all future activities in the project
- To monitor the remaining critical risks and estimate their impact on the project and its business case.
- To specify additional aspects of the project plan.

- These objectives are intended to get the project cycle to the final objective, completing the business case including: providing an
  - accurate estimate of the benefits, costs, and ROI of the project, and
  - a useful plan for the construction and transition phases of the project.

The activities needed to accomplish these objectives are described below.

(1) Revise the plan of the elaboration phase. The plan developed in the inception phase should be revised or re-worked in light of the newly available resources of the elaboration phase.

(2) Build the team necessary to carry out the elaboration phase. The team assembled for the inception phase is carried over as much as possible to this phase. In addition new members are added with skills such as structural, measurement, and causal modeling, necessary to provide continuity into the construction phase.

(3) Determine evaluation criteria. The evaluation criteria relate to four areas: extending the requirements, establishing the baseline architecture, mitigating critical risks, and assessing the business case. First, in relation to extending

the requirements one must assess whether the task patterns, agents, and requirements needed for the architectural baseline and the final business plan have been specified. Second, will the "to-be" baseline architecture of the KM solution support the requirements, other needs felt by stakeholders, the construction phase of the project, and the addition of new features that later versions of the solution may introduce? Third, are the critical risks mitigated adequately, or planned for, or clearly removable during the construction phase? Have they all been identified and analyzed sufficiently to support the elaboration phase business plan?

(4) Define the iterations of the elaboration phase. Elaboration may involve more than one iteration at the outset. Inception may have provided an understanding of the task patterns suggesting that more than a single iteration will be needed to complete the business case, and also suggesting a division of the work by task patterns, task pattern realizations and architectural subsystems. If so this is the place to make such a decision and to define multiple iterations.

(5) Execute the core work flows of the elaboration iteration. The elaboration phase iteration(s) will focuses equally on strategic planning and requirements capture, analysis, and design and less so on testing and implementing. In the design phase a prototype of some of the most significant task patterns is much more likely to be developed than in the inception phase. The prototype will likely be developed for KM personnel themselves, rather than exposing the new solution to other stakeholders at this early stage.

In requirements capture activities include: finding additional agents and task patterns for the KM solution, prioritizing task patterns, detailing some fraction of the task patterns to further specify the architecture of the solution, capturing additional non-functional requirements necessary for the solution, and structuring (simplifying, synthesizing and organizing) the task pattern model. In the elaboration phase, the task patterns, with few exceptions must be identified, described, and in the case of all architecturally significant task patterns, analyzed. This is necessary to ensure that managers are aware of all architectural risks that can impact the project.

In analysis in the elaboration phase, the analysis model begun in inception is extended to support the related tasks of specifying task patterns and the solutions architecture in detail for all of the architecturally significant task patterns identified in requirements capture. This involves architectural analysis that extends the architecture to the baseline necessary to support the KM solution. It also involves detailed analysis of the architecturally significant classes in the object model, and grouping of these classes into clusters that collaborate to perform services. In UML [19], these are called service packages.

In the design work flow, the design model begun in inception is extended to further tie down the physical architecture of the KM solution in the critical areas covered by the architecturally significant task patterns analyzed in inception and the analysis work flow of the elaboration phase. This is architectural design. Its objective is to design the architecturally significant task patterns, classes, and subsystems.

The architect here identifies the layered architecture including the business structures that must collaborate to perform the task patterns. The architect then identifies the subsystems of collaborating business structures and the communications interfaces used by these structures. Thus, the architect continues to construct a structural model of the target system. Next, the architecturally significant analysis classes are translated into design classes. Then the network of business structures and their communication patterns is designed. Next, each of the architecturally significant task patterns is now described in detail, resulting in a set of diagrammed task pattern realizations corresponding to the architecturally significant task patterns. The architecturally significant classes represented in the task pattern realizations are also designed at this point, including the measurement models that are methods in some of the classes, as are the subsystems of related design classes that collaborate in the task pattern realizations. This completes the design work. It will have produced an architecture, architecturally significant task patterns, task pattern realizations, design classes including measurement models, and subsystems including structural models, all sufficient to allow completion of the business plan for this iteration of the project.

Implementing and testing in the elaboration phase focuses on architecture and on the architecturally significant business structures of the "to-be" model. This also includes implementing and testing design classes and subsystems, and integrating the architecture into the final baseline necessary to support the business case. This implementation and testing work involves computer simulation of the architecture, task patterns, task pattern realizations and other aspects of the system. So, the first implementation of the architecture is the implementation of a simulation representing an approximation to the solution for the architecturally significant task patterns.

As part of implementing and testing, it may also be necessary to produce a prototype or pilot of the operation of key task patterns in the KM solution. To do this, it is safest to use KM-level knowledge workers to test the innovation represented by the KM solution. In the elaboration phase however, this should be a scaled down pilot, and should involve only enough task patterns to support the business case for the solution.

(6) Complete the Business Plan. The architectural baseline and other results of the core elaboration work flow, provide sufficient information to support a decision on whether or not to implement the project, and an accurate analysis of the prospects for success in its construction and transition phases. By this point, the project personnel have done their best to analyze and mitigate risks and to get a very firm initial idea of what the KM solution will look like and how it will work The business plan should provide hard estimates of schedules, effort, and cost, and also a much more precise estimate of anticipated return on investment from the KM solution.

(7) Assess the Iterations of the elaboration phase. Here the project team and the project stakeholders match the results of iterations against the evaluation criteria specified at the beginning of the elaboration phase. Where more than one iteration has occurred, the existence of multiple increments and milestones has served to keep stakeholders expectations about the elaboration phase in check. When the final iteration is completed, they should be in agreement with the project team's assessment of whether the iterations have been successful and whether the project should move forward into the construction phase.

(8) Plan the Construction Phase. Toward the end of the elaboration phase, it is necessary to begin to plan the construction phase. Important considerations are the number of iterations in the construction phase, the remaining project risks, and the order of task pattern realization and subsystem implementation. The number of iterations depends on the size and complexity of the KM solution. If a complex solution involving many task patterns and multiple subsystems is involved, it will make sense to add and structure iterations based on implementing related task patterns and subsystems. Plan to investigate and mitigate remaining risks before each risk becomes operative in construction. Finally, the order of implementation of parts of the solution should be guided by the relationship of these parts to one another. This comes down to noting how business structures and subsystems are related and implementing related subsystems together.

Construction

Construction is the phase where the solution is developed from a stable architectural baseline to the full business structure configuration specified by the solution. Construction will also generally involve creation of new task patterns and further development of architecture to fill in the gaps left by the first two phases. But the primary objective of construction is to implement a large-scale computer model that will represent the KM solution and allow for its testing, transition, and ultimately continuous assessment and evaluation. Implementing this model is likely to be an incremental process involving multiple iterations. Construction involves the following activities.

(1) Replan the construction phase. This activity is made necessary because approval of the business plan formulated in the elaboration phase may well occur with changes, most probably reductions in the resources allocated for the project cycle. The plan will therefore have to be adjusted for these changes including planning new iterations.

(2) Building the team for the construction phase. To the extent possible the elaboration phase team should be retained, particularly its design. The size of the team is normally expanded considerably in this phase, but the size of the expansion and the composition of the team depends on the type of KM solution to be implemented. Certainly, the programming staff oriented toward simulation and artificial intelligence disciplines, as well as staff experienced in measurement, structural, and causal modeling must be greatly augmented. In addition, however, staff skilled at executing KM solutions in the transition phase must be added during this phase to socialize them into the project and prepare them for the transition phase.

(3) Develop the evaluation criteria. Mostly these criteria are already set by the task patterns and the non-functional requirements. That is, the construction phase is to be evaluated based on the degree to which it fulfills both the task patterns and non-functional requirements. The construction phase also involves preparing documentation and instructional materials for the KM solution. Evaluation criteria must be developed for these as well.

(4) Execute the core work flows of the iterations of the construction phase. In requirements capture we again find new agents and task patterns, not introduced earlier, and then go on to prioritize, detail them, and structure them. In this construction phase we complete requirements capture for all task patterns and all non-functional requirements in the system. In analysis, we also complete the analysis model including the architectural analysis, the task pattern model, the object and object interaction models, and the analysis packages.

In design, the remaining task patterns (the non-architecturally significant ones), that is, most of them, are completely designed. The architectural design is also completed here, including the subsystem design, and the design of non-functional requirements. Much of design is concerned with specifying the design classes and the object model relating these. Once again the activities of measurement, structural, and causal modeling are essential to design and creation of the physical realization of the object model.

The implementation work flow in this phase represents a major portion of the effort in each iteration of the construction phase. It is directed at physically implementing the design in a working computer simulation model. This

involves implementing the architecture, the classes and subsystems of the object model, including measurement relations, structure, and dynamics, performing unit testing on model components, and integrating these components into a model of the KM solution. In addition to the preferred model of the solution KMFM requires that alternative models be formulated. This is necessary to provide for fair comparison testing of the model underlying the KM solution.

In addition to implementing a computer model of the KM solution, part of that solution may require implementing an Information Technology application: a document management application, or a collaborative application, or an Enterprise Knowledge Portal. If this is the case, The implementation work flow will require implementing a software development methodology such as RUP.

The testing work flow is the other major work flow of the construction phase. It includes activities such as planning tests, designing tests, performing integration tests, performing system tests and evaluating tests. Among the integration, system, and evaluating tests are tests of the measurement, structural, causal, and dynamic models embedded in the computer simulation model. These tests will make use of alternative models of the system, empirical data available in the enterprise, as well as judgmental data created for the purpose of testing the computer model. The testing activities may also involve data mining activities designed to validate various aspects of the computer model.

(5) Manage and control the project. The construction phase involves a number of iterations and resulting increments of the solution. The iterations implement the task patterns iteratively and incrementally until the whole solution is delivered. With each iteration more and more of the system is implemented and then tested. Through this process, managers monitor and assess whether the goals of each iteration are being met. When variance occurs, managers can take corrective action. If variances continue, stakeholders are informed and adjustments to the project plan are proposed and perhaps accepted.

(6) Assess the iterations and the construction phase. This includes reviews of performance compared to the plan, "updating the risk list," [7, P. 393] revising the plan for future iterations, determining whether the construction phase has accomplished its goals and whether the KM solution is ready for transition.

(7) Planning the transition phase. The construction phase produces a physical implementation of the KM solution in the form of a simulation, along with a working software application if that is part of the solution. The transition phase is the phase where the solution is tested, evaluated and deployed by the agent community. The plan for such tests, evaluations, and deployments is

the last activity of the construction phase. There is not formula for planning the transition phase. Details of the plan will vary with the KM solution developed in other phases.

Transition

Activities in the transition phase are as follows.

(1) Build the transition phase team. This team is different from the team in earlier phases. The objectives of the transition phases are to implement the policies, procedures and activities specified in the KM solution. Developing the solution involved a lot of effort from information technology personnel either to implement the computer simulation aspect of the project or to implement the part of the solution that was an IT application. In the transition phase there is less need for programming or design staff. Only a small number of IT staff continue to be necessary to use the solutions model to evaluate and analyze the impact of implementing the solution. The rest of the transition staff is involved in communicating the solution, helping to promulgate new policies at the heart of the solution, and actually helping with implementing it in the field by helping to organize communities of practice, or training people in the solution, or performing particular kinds of analyses prescribed by the solution, or leading group decision processes that may be part of the solution.

(2) Determine the evaluation criteria. Some of the evaluation criteria are given by the original goals and objectives set for the project. Does the KM solution promise to close the benefit gap that originally provided part of the justification for the project. Other evaluation criteria are provided by the task patterns themselves. That is, the solution in the transition must work in that stakeholder agents must be able to perform the task patterns they originally requested and, with enough frequency to satisfy them, get the results they anticipate. Where it is too soon to determine whether the solution closes the benefit gap, the second set of criteria are very important, because the stakeholder agents assume that if the system works as originally envisioned, it will, ultimately, close the benefit gap.

(3) Execute the transition phase work flow. This work flow is very different from the other phases. It involves decision and action on the part of the KM team and stakeholders involved in the solution. One of the initial activities is training knowledge workers to use the solution. There is also KM planning using the solution, deciding on and implementing activities according to the plan, and monitoring and evaluating the results.

Of all the core work flows therefore, the one used most in the transition phase is the testing work flow including monitoring and evaluating. The testing work flow is a constant throughout the life of the solution. Other activities in the

transition work flow will depend on the nature of the KM solution at issue. Activities will differ if the solution being implemented is a KM program, an Enterprise Knowledge Portal, a community of practice, a web-enabled data warehousing application, a new collaborative process for supporting innovation, etc. Activities to be implemented will be specified in the transition plan.

(4) Manage and control the transition phase and the project plan. During the transition phase the project manager tracks performance against the planned schedule, effort and costs. Variances require corrective actions and adjustments to the schedule. At the end of the phase a review is performed to compare performance against plans.

(5) Assess the iterations and the transition phase. This assessment is the major effort at evaluating whether the project has met its goals and recording lessons learned for future efforts. Decisions are made about whether further iterations are needed to complete the transition phase.

(6) Plan the next generation of the KM solution. Iterations and phases occur within the confines of a project cycle. The cycle, in turn, is part of a long-term pattern of solution development and implementation, composed of many cycles, iterations, and work flows. The end of the transition phase is the end of a cycle, but not necessarily the end of the project or the evolution of the solution. This last activity in the transition phase is a recognition that the search for an adequate solution to a problem motivating a KM solution may continue even though a solution is found and implemented. No KM solution is perfect. There is always room for improvement. The question is whether a business case can be made for the next generation plan.

In describing the above phases of development cycles, the steps in each work flow have been described in sequential fashion. This was done for convenience of exposition. In fact, the steps in each work flow can and are frequently performed concurrently, in a sequence that is most efficient in a project context. Iterations may also be performed concurrently. The general point is that KMFM is not a linear life cycle methodology either at the level of the iteration or within iterations. The patterns of development in it vary from cycle to cycle and follow a logic specific to the context of the cycle. The development pattern involves feedback and concurrent engineering. It is non-linear in character and is not a life cycle development methodology in the classical sense.

Between Development Cycles: The Maintenance and Enhancement Phase

It is rare that a development cycle is immediately succeeded by another development cycle. It takes time and "politicking" for the next generation solution to receive both moral and financial support. During the period between

development cycles, maintenance and enhancement activities occur in iterations in the maintenance phase. Each iteration follows the pattern of the maintenance and enhancement work flow specified above. The phase begins with continuous monitoring and evaluation of the performance of the KM solution. The steps in the phase, once again, are as follows

Step 1: Use the results of Testing (Monitoring and Evaluating) to redesign the KM Solution by Iterating over the design work flow.

Step 2: Change the KM Solution to Correspond to the Redesigned Model and Solution.

Step 3: Implement the Redesigned Solution.

Step 4: Test the Redesigned Solution

Step 5: Go Back to Step 1

While this maintenance and enhancement work flow allows a KM solution to evolve continuously, its adaptive capability is limited by the fact that it takes the design workflow as the basis for change in the system. As a system evolves, new needs occur that can only be handled by the next generation of the proposed KM solution. When this is made clear through an appropriate business case, the next generation system can receive funding and a new development cycle can begin.

Issues

KMFM and UML-based Methodology

KMFM borrows liberally from the Rational Unified Process [7], but is different in a number of ways. First, KMFM specifies functional requirements through non-determinate task patterns. This is a fundamental difference recognizing that KM system solutions are non-mechanical in nature and, at best, facilitate desired system behavior and goal attainment by reinforcing self-organizing tendencies in the enterprise.

Second, while RUP focuses on software application systems and their modeling and implementation, KMFM focuses on modeling enterprise systems and KM solutions, and on implementing and evaluating such solutions. That is, KMFM is concerned with measurement and causal modeling, analysis of complex adaptive systems and modeling of the behavior of natural intelligent agents.

Third, the substantive concerns of KMFM are very different from RUP. KMFM is a methodology for developing KM solutions for enterprise systems. That is, it is a social systems methodology. As such its activities go beyond simple object and

state transition modeling to concerns such as analytic hierarchy development, non-monetary benefit cost modeling, statistical modeling and multivariate analysis, neural network modeling and simulation, semantic network analysis, and many other forms of analysis not normally part of RUP activities. Most importantly, KMFM is concerned with modeling the process of knowledge claim validation itself.

Fourth, KMFM is a superset of RUP, or other UML-based methodologies in the sense that the software development methodolgies are part of the larger set of KMFM activities.

KMFM Benefit Estimation

KMFM's benefit estimation task is based on a more specialized methodology for benefit estimation developed earlier. It uses measurement modeling methods, ratio scale estimation methods, analytical hierarchy modeling and other techniques to arrive at a methodology for benefit-cost estimates that incorporate and transcend monetary measures of benefit. The methodology is described in detail in a chapter of a longer industry report. [15] A revised version is forthcoming in an article in the April issue of Knowledge and Innovation. [32]

Tools

Every methodology has tools associated with it. Space considerations prevent a discussion of tools here, though the article by Barquin [33] in this issue contains a brief KM tool survey. But it may be helpful to name some of the major tools useful in applying KMFM. These include: Knowledge Processing tools and methods, and also software tools.

- Knowledge processing tools and methods include: communities of practice, story-telling, Group Decision Process Methods (such as Delphi Technique, Nominal Group Technique, Group Value Measurement Technique, Knowledge Café, Team Analytic Hierarchy Process, Joint Requirements Modeling, and Joint Application Design), cultural analysis, value network analysis, object modeling, causal modeling, neural network modeling, fuzzy systems modeling, Bayesian Belief Networks, influence network analysis, semantic network analysis, genetic algorithms and programming, process modeling, measurement modeling, systems modeling, system dynamics, balanced scorecard modeling, intangible asset analysis, and enterprise performance measurement.

- Software tools include: Enterprise Information Portals, Enterprise Knowledge Portals, Intelligent Software Agents, XML and enhanced XML formats such as XML topic maps (XTM), and Scalable Vector Graphics (SVG), Stateless Application Servers Application servers and Business process Engines, full-

text indexing, content management, ROLAP and MOLAP Servers, Business Intelligence tools, Data Warehouses, Data Marts, Operational Data Stores, Knowledge Discovery in Databases/Data Mining tools, Data Staging areas, ETL Servers, Collaboration tools, Group Decision Process Tools, Analytical Modeling and Simulation tools, Computer-assisted learning, Semantic Network Analysis tools, text abstracting and full-text indexing, querying and reporting, searching/retrieving, Packaged Analytical Applications, Balanced Scorecard applications, Object Modeling and related tools, Expert Assessment Capture (EAC), Content Publication and Broadcasting, and Personal Knowledge Management.

The above lists are far from complete, but they make the point that a survey of tools and methods accompanying KMFM would be a rich analytical endeavor.

Conclusion

This paper is the first presentation of a new methodology for developing Knowledge Management solutions called Knowledge Management Framework Methodology (KMFM). The methodology is task pattern-driven, business structure centric, iterative and incremental. It distinguishes development and maintenance cycles, phases within each cycle, iterations within each phase, and work flows within each iteration. Generally, many of the steps within each work flow may be implemented concurrently, as can many of the iterations within each phase. Work flows are not restricted to particular phases but are spread across multiple phases.

The effect of all this is a comprehensive, but flexible and continuous process of KM solutions development. The process is focused on creating computer simulation models of the KM solutions being formulated. These models then serve as a backdrop for field implementation of the solutions and as a continuing tool for policy and planning guidance, and for monitoring and evaluating the impact of the KM solution. The methodology provides a place for process tools and IT tools. It incorporates conceptual frameworks such as the KMCI's knowledge Life cycle (KLC) and Metaprise models. It also provides methods for measuring the balance of benefits to costs resulting from KM solutions, as well as methods for developing KM metrics.

References

[1] See Yogesh Malhotra's compilation at http://www.brint.com

[2] Joseph M. Firestone, "Knowledge Management: A Framework for Analysis and Measurement," White Paper Prepared for Executive Information Systems, Inc., Wilmington, DE, October, 2000, available at: http://www.dkms.com/White_Papers.htm..

[3] Joseph M. Firestone, "Enterprise Knowledge Management Modeling and Distributed Knowledge Management Systems," White Paper Prepared for Executive Information Systems, Inc., Wilmington, DE, January 1999, available at: http://www.dkms.com/White_Papers.htm.

[4] Alistair Cockburn, Surviving Object-oriented Projects: A Manager's Guide (Reading, MA: Addison-Wesley, 1998).

[5] Ivar Jacobson, Object-oriented Software Engineering (Reading, MA: Addison-Wesley, 1998).

[6] James Rumbaugh, Ivar Jacobson, and Grady Booch, The Unified Modeling Language Reference Manual (Reading, MA: Addison-Wesley, 1998).

[7] Ivar Jacobson, Grady Booch, and James Rumbaugh, The Unified Software Development Process (Reading, MA: Addison-Wesley, 1999).

[8] Thomas L. Saaty, The Analytic Hierarchy Process (Pittsburgh, PA: RWS Publications, 1990),

[9] Joseph M. Firestone, "Accelerated Innovation and KM Impact," Financial Knowledge Management, 1, no. 1 (1999), 54-60, also available at http://www.dkms.com.

[10] Joseph M. Firestone, "Enterprise Knowledge portals: What they are and What They Do", Knowledge and Innovation: Journal of the KMCI, 1, no. 1 (2000) 85-108.

[11] Mark McElroy, "The Second Generation of KM," Knowledge Management (October, 1999), Pp. 86-88, also available at http://www.macroinnovation.com.

[12] Steven Cavaleri and Fred Reed, "Designing Knowledge Creating Processes," Knowledge and Innovation: Journal of the KMCI, 1, no. 1 (2000) 109-131.

[13] Edward Swanstrom, Joseph M. Firestone, Mark W. McElroy, Douglas T. Weidner, and Steve Cavaleri, "The Age of The Metaprise," (revised version) Knowledge Management Consortium International, Gaithersburg, MD, 1999, available from the author.

[14] Joseph M. Firestone, "The Metaprise, The AKMS, and The Enterprise Knowledge Portal," Working Paper Prepared for Executive Information Systems, Inc., Wilmington, DE, March 1999, available at: http://www.dkms.com/White_Papers.htm.

[15] Joseph M. Firestone, Approaching Enterprise Information Portals (Wilmington, DE: Executive Information Systems, Inc., 1999) available at http://www.dkms.com/EIPMarketing.htm.

[16] Mark W. McElroy, "The New Knowledge Management," Knowledge and Innovation: Journal of the KMCI, 1, no. 1 (2000) 43-67.

[17] http://www.macroinnovation.com.

[18] John H. Holland, Hidden Order (Reading, Mass.: Addison-Wesley, 1995).

[19] Hans-Erik Eriksson and Magnus Penker, UML Toolkit (New York, NY: John Wiley and Sons, 1998).

[20] Joseph M. Firestone, "Remarks on Concept Formation: Theory Building and Theory Testing," Philosophy of Science, 38, no. 4 (1971) 570-604.

[21] Joseph M. Firestone and Richard W. Chadwick, "A New Procedure for Constructing Measurement Models of Ratio Scale Concepts," International journal of General Systems, 2 (1975), 35-53.

[22] Joseph M. Firestone, "Knowledge Management Metrics Development: A Technical Approach," White Paper Prepared for Executive Information Systems, Inc., Wilmington, DE, June 1998, 33 Pp. Available at: http://www.dkms.com/White_Papers.htm.

[23] Joseph M. Firestone, "Manhattan Projects for the Study of Foreign Policy," Fields Within Fields: Journal of the World Institute, No. 13 (1974), 73-80.

[24] Herbert A. Simon, Models of Man (New York, NY: John Wiley and Sons, 1957).

[25] Carl F. Christ, Econometric Models and Methods (New York, NY: John Wiley and Sons, 1966).

[26] P. M. Bentler, and D. J. Weeks, "Multivariate Analysis with Latent Variables: Causal Modeling," Annual Review of Psychology, 31 (1980), 419-456.

[27] Kenneth A. Bollen, Structural Equations with Latent Variables (New York, NY: John Wiley and Sons, 1989).

[28] Peter Senge, The Fifth Discipline (New York, NY: Doubleday/Currency, 1990).

[29] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen, Object-Oriented Modeling and Design (Englewood Cliffs, N.J.: Prentice-Hall, 1991).

[30] Kent Beck, Extreme Programming (Reading, MA: Addison-Wesley, 1999).

[31] Alistair Cockburn, "Crystal 'Clear:' A Human-powered Software Development Methodology for Small Teams, (2001) available at http://members/aol.com/humansandt/crystal/clear/.

[32] Joseph M. Firestone, "Estimating Benefits of Knowledge Management Initiatives: Concepts, Methodology, and Tools," Knowledge and Innovation: Journal of the KMCI, 1, no. 3 (forthcoming, 2001).

[33] Ramon Barquin, "What is Knowledge Management," Knowledge and Innovation: Journal of the KMCI, 1, no. 2 (2001) .

Biography

Joseph M. Firestone, Ph.D. is Vice-President and Chief Scientist of Executive Information Systems (EIS), Inc. Joe has varied experience in consulting, management, information technology, decision support, and social systems analysis. Currently, he focuses on product, methodology, architecture, and solutions development in Enterprise Information and knowledge Portals, where he performs Knowledge and knowledge management audits, training, and facilitative systems planning, requirements capture, analysis, and design. Joe was the first to define and specify the Enterprise Knowledge Portal Concept. He is widely published in the areas of Decision Support (especially Enterprise Information and Knowledge Portals, Data Warehouses/Data Marts, and Data Mining), and Knowledge Management, and has recently completed a full-length industry report entitled "Approaching Enterprise Information Portals."

Joe is a founding member of the Knowledge Management Consortium International (KMCI), a member of its: Executive Committee, Board of Directors, Metaprise Project, and Governing Council of the KMCI Institute. He is also the Director of the KMCI Research Center, Editor of the new journal "Knowledge and Innovation: Journal of the KMCI," and Chairperson of the KMCI's Artificial Knowledge Management Systems Special Interest Group, Joe is a frequent speaker at national conferences on KM and Portals. He is also developer of the web site www.dkms.com, one of the most widely visited web sites in the Portal and KM fields. DKMS.com has now reached a visitation rate of 110,000 visits annually.